DECnet–ULTRIX

digital

Network Management

Order Number: AA–EE38D–TE

DECnet–ULTRIX

Network Management

May 1990

This manual introduces DECnet databases and components and shows how you use the Network Control Program (ncp) to configure, monitor, and test the components.

Supersession/Update Information:	This is a revised manual.
Operating System and Version:	ULTRIX V4.0
Software Version:	DECnet–ULTRIX V4.0

digital™

Order Number: AA–EE38D–TE

AA–EE38D–TE
May 1990

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Restricted Rights: Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227–7013.

The following are trademarks of Digital Equipment Corporation:

DEC	PDP	VAX
DECnet	ULTRIX	VMS
DECUS	UNIBUS	digital™

UNIX is a registered trademark of AT&T in the USA and other countries.

This manual was produced by Networks and Communications Publications.

Contents

Figures

Tables

Preface

This manual shows you how to manage a DECnet–ULTRIX node in the DECnet environment. DECnet–ULTRIX software works with systems running ULTRIX software and conforms to the Digital Network Architecture (DNA). DNA, the model for all DECnet implementations, allows all Digital operating systems to participate in the same network.

Manual Objectives

This manual describes the network databases and components and shows you how to configure, monitor, and test network components using the Network Control Program (NCP). For detailed descriptions of the **ncp** commands and error messages, refer to the *DECnet-ULTRIX NCP CommandReference* manual. Other topics include loopback testing, event logging, and displaying system counters and database information.

Intended Audience

This manual is for the network manager, the person responsible for configuring, managing, and maintaining the network.

Structure of This Manual

This manual contains six chapters:

Chapter 1	Gives an overview of DECnet–ULTRIX network management. Introduces the configuration databases and the tools for managing them.
Chapter 2	Describes basic network components and tells you how to use **ncp** commands to configure them.
Chapter 3	Explains how to control access by using access-control information and setting up proxy.
Chapter 4	Tells how to monitor the network using display commands, network counters, event logging, and object log files.
Chapter 5	Describes node-level and circuit-level loopback tests, and **dts/dtr** tests.
Chapter 6	Contains quick-reference manual pages for the DECnet–ULTRIX network maintenance commands **ncp** and **dts/dtr**.

Related Documents

In addition to this manual you may want to refer to the following DECnet–ULTRIX documents:

- *DECnet–ULTRIX Release Notes*

  Contains information and updates not included in the DECnet–ULTRIX documentation set.

- *DECnet–ULTRIX Installation*

  Contains step-by-step procedures for installing DECnet–ULTRIX software.

- *DECnet–ULTRIX NCP Command Reference*

  Describes the **ncp** commands you use to define, monitor, and test network components.

- *DECnet–ULTRIX Use*

  Describes the DECnet–ULTRIX user commands and shows how you use them to perform file transfer and other user tasks.

- *DECnet–ULTRIX Programming*

  Describes the DECnet–ULTRIX system calls, subroutines, and application programming procedures.

- *DECnet–ULTRIX DECnet-Internet Gateway Use and Management*

  Describes the DECnet-Internet Gateway and explains how you install, use, and manage it.

For a detailed description of the Digital Network Architecture, refer to the *DECnet Digital Network Architecture (Phase IV), General Description.*

Acronyms

The following acronyms appear in this manual:

DAP	Data Access Protocol
DDCMP	Digital Data Communications Message Protocol
DNA	Digital Network Architecture
dtr	DECnet Test Receiver
dts	DECnet Test Sender
ECL	End Communication Layer
evl	Event Logger
fal	File Access Listener
mir	Loopback Mirror
ncp	Network Control Program
nml	Network Management Listener
NSP	Network Services Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol

Terminology

In this manual, an *adjacent node* is a node connected to the local node by a single physical line.

The following terms are used interchangeably in this manual:

- Running database and operational database.
- Sink node and logging host.

Graphic Conventions

This manual uses the following conventions:

Convention	Meaning
`Example`	Examples appear in this type. Red type in examples indicates literal user input.
special	All commands and parameters appear in **special type**.
lowercase	If a command appears in lowercase type in a command format or in an example, you must enter it in lowercase.
italics	Indicates a variable, for which either you or the system must supply a value.
{ }	Indicate that you must specify one of the enclosed options, but no more than one. Do not type the braces when you enter the command.
[ ]	Indicate that you can use one, and only one, of the enclosed options. Do not type the brackets when you enter the command.
( )	Parentheses enclose a set of options that you must specify together or not at all.
...	Indicates that the preceding item can be repeated one or more times.
Vertical list of options	A vertical list of options—not enclosed within braces, brackets, or parentheses—indicates that you can specify any number of options or none at all.
key	This is a symbol for a key on the terminal keyboard. CTRL/*key* represents a CTRL key sequence, where you press the CTRL key at the same time as the specified key.

All Ethernet addresses are hexadecimal; all other numbers are decimal unless otherwise noted.

Chapter 1

Overview of the DECnet Network

Before you perform network management tasks, you should already be familiar with the DECnet network components, configuration databases, and network management tools.

This chapter briefly overviews DECnet structure, components, and configuration databases. Other topics include tools to configure and control the network, and basic network management tasks.

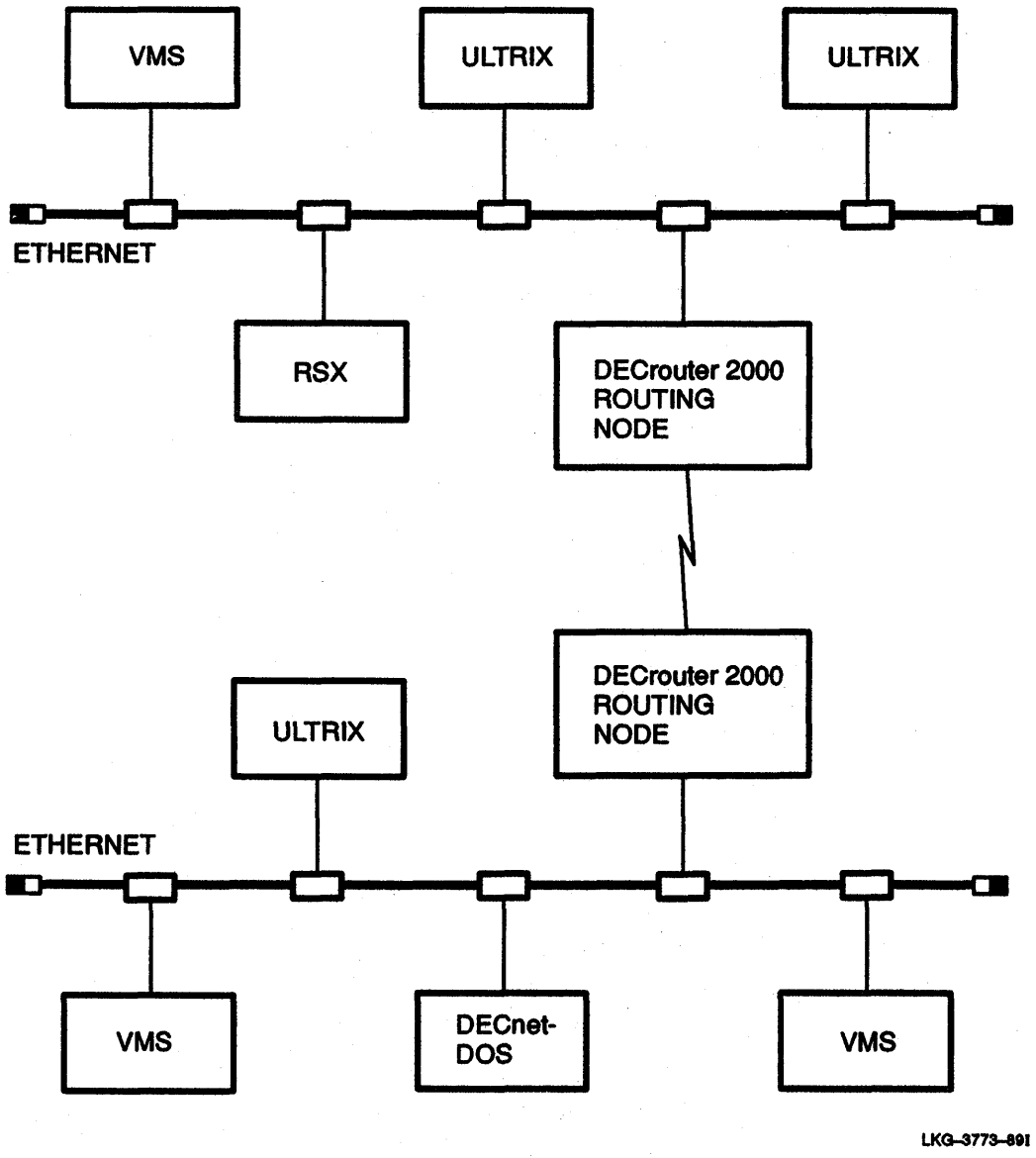
1.1 DECnet and the ULTRIX Operating System

DECnet is the name for all of the software and hardware products that enable Digital operating systems to participate in a DECnet network. The DECnet–ULTRIX Phase IV software enables an ULTRIX operating system to function as an end node on a DECnet network.

As an interface to an ULTRIX system, DECnet–ULTRIX software supports the protocols necessary for communicating over the network and the functions necessary for configuring, controlling, and monitoring nodes. As a DNA Phase IV end node, a DECnet–ULTRIX node supports connections to other systems on an Ethernet or to a single system on a DDCMP point-to-point line. Routing nodes provide access to systems beyond your local Ethernet or DDCMP point-to-point line.

Figure 1-1 shows a diagram of a configuration that includes two Ethernets with a variety of systems that communicate with each other by way of routing nodes.

Figure 1–1: Sample DECnet–ULTRIX Ethernet Configuration



1.2 Network Components

DECnet network components are defined as follows:

node

A node is a computer system that runs DECnet software as an interface between its operating system and the network. A node can process, send, and receive network information.

Network terminology includes three terms for describing nodes. These terms reflect your relationship to the node:

- **Local node.** Your node—the node from which you operate.
- **Remote node.** Any node other than yours.
- **Executor node.** The node where your current network management command executes.

Object

An object is a DECnet Application layer process that you can connect to over a logical link to perform general-purpose network services.

Line

A line is the physical connection between nodes. DECnet–ULTRIX software supports connections between nodes on two types of lines: DDCMP point-to-point and Ethernet. A DDCMP point-to-point line links two nodes directly; an Ethernet line links all nodes to a common media, such as a coaxial cable. Each node can access the line by means of a circuit.

DDCMP point-to-point nodes are connected to the line by DMV or DMC controllers. Ethernet nodes are connected to the line by communications controllers. (See the software product description (SPD) for a list of controllers.)

Circuit

A circuit is a logical connection that carries information between adjacent nodes and operates over the physical line. A circuit can be identical to a physical link or can be multiplexed with many other circuits on one line.

A DDCMP point-to-point circuit is the logical means for connecting your DECnet–ULTRIX node to one adjacent node on the network. Usually the adjacent node is a DECrouter in the same area. An Ethernet circuit enables you to connect to all other nodes on an Ethernet cable. Each node on an Ethernet cable is considered to be adjacent to every other node on that cable.

1.3 Network Management Tools

The DECnet–ULTRIX software offers three network management tools: the Network Control Program (**ncp**), the Event Logger (**evl**) and the **mop_mom** utility.

1.3.1 The Network Control Program (ncp)

The Network Control Program is a utility program that lets you manage the components of your network and test network performance. It also lets you display information on the condition, characteristics, and performance of network components.

All DECnet systems share a common set of commands and parameters for network management. In addition, many DECnet systems have system-specific commands and parameters. For example, DECnet–ULTRIX has system-specific commands for objects.

1.3.2 The Event Logger(evl)

The Event Logger logs network events so that you can monitor network activity. Certain **ncp** commands let you specify whether event messages are to be logged to one or all of the following sink node devices:

- Console — Any device that receives and records event messages.
- File — A user-specified file that receives event messages.
- Monitor — A user-supplied program that receives and processes events.

Some events that the Event Logger record include:

- Changes in node state.
- Passive loopback (when the executor is looping back circuit test messages).
- Line and node counter activity.

This type of information can be useful in maintaining and tuning the network, because the Event Logger can record it continuously.

See Chapter 4 for event-logging procedures.

1.3.3 The mop_mom Utility

The **mop_mom** utility lets you perform down-line loading and up-line dumping tasks. It spawns a **mop_mom** daemon that listens for down-line load and up-line dump requests on behalf of the local ULTRIX node.

When a down-line load or up-line dump request is received from a target node, the **mop_mom** utility spawns the **mop_dumpload** loader to process the load request.

For more detailed information about the **mop_mom** command, see the **mop_mom(8)** description in the *ULTRIX Command Reference* manual.

1.4 Configuration Databases

The configuration databases provide the information that a DECnet–ULTRIX node needs to function as part of a network. DECnet–ULTRIX software uses two databases: the permanent database and the volatile (running) database. You use **ncp** commands to configure network components in both databases.

During the installation of the DECnet–ULTRIX software, the installer defines the following network components in the permanent database:

- The local node and at least one other node in the network
- The local physical line
- The local circuit
- Each logging sink for the event logger

- Each object known to the network and to the local node

1.4.1 Displaying Volatile and Permanent Databases

Before you modify a network component, you must use a different command to display its parameter values, depending on its location. If it is in the permanent database, use the **list** command to display parameter values for each component. If it is in the volatile database, use the **show** command.

See Chapter 4 for examples of how to display information about each component.

1.4.2 Modifying Volatile and Permanent Databases

You use **ncp** commands to modify parameter values defined in either database for any of these network components. You use different commands to configure network components in the volatile database than in the permanent database.

The **define** and **purge** commands act on the permanent database and take effect when the network is restarted, whereas their functional counterparts, the **set** and **clear** commands, act immediately on the volatile (running) database.

To modify values in the permanent database, use the following commands:

define	adds or changes a parameter value.
purge	removes parameter values.

To modify values in the volatile database, use the following commands:

set	changes or resets a parameter value.
clear	removes parameter values.

Changes you make to the volatile database go into effect immediately. When you start up the system, the initial version of the volatile database is a copy of the permanent database.

If you alter the volatile database, you can restore the parameter values from the permanent database for any component by using the **all** parameter with a **set** command for that component. To display the current parameter values in the volatile database, use the **show** command.

Changes you make using the **set** and **clear** commands remain in effect until you make another change, restart DECnet–ULTRIX software on the local system, or reboot the operating system. When you reload, the system configuration matches the parameter values in the permanent database.

1.5 Privileges

On ULTRIX systems, you need superuser privileges to execute **ncp** commands that change a database. Other Digital systems may also require privileges for the same **ncp** commands. To determine the privileges you need to issue **ncp** commands at a DECnet–ULTRIX node for remote execution, see the network management documentation for that remote node.

1.6 Remote Access

You can issue **ncp** commands for execution at either the local node or a remote node. If a remote non-ULTRIX node supports an expanded set of **ncp** commands, you can execute the full set of **ncp** commands on that node. (See the remote system's documentation for a list of supported commands.)

For more information about remote access, see the *DECnet–ULTRIX NCP Command Reference* manual.

1.7 Down-Line Loading a Remote Node

The Network Control Program gives you down-line loading capabilities so that you can boot a remote node from your local node. You can down-line load a remote node from your local node by issuing **ncp load** and **trigger** commands.

You can also specify parameters on the **ncp load node** and **ncp trigger node** command lines to override current parameter values in the load host's database.

NOTE

Before you can use the **ncp load** and **trigger** commands, the ULTRIX **mop_mom** utility must be installed on your system. For further information, see **mop_mom** in the ULTRIX documentation set.

The following sections explain how to use **load** and **trigger** commands to initiate a down-line load from a load host.

1.7.1 Using the ncp load Command

If you use the **ncp load** command, you must issue it at a load host. The **ncp load** command ensures that the load host at which you issue the command is the node that performs the down-line load.

After you issue the **load** command on the load host, the down-line load proceeds as follows:

1. The load host sends a **mop remote console boot** message with a direct load option specified.
2. When the remote node receives this message, it sends a **mop request program** message directly to that load host.
3. The load host and the remote node use additional **mop** messages to transfer the remote node's software image into the remote node's memory.

1.7.2 Using the trigger Command

If you use the **ncp trigger** command, you can issue it from any supported network management host. Because it does not have to wait for a specific load host to respond to a load request, the **trigger** command is usually faster than the **load** command. The **trigger** command does not specify which load host performs the down-line load. Therefore, you may not know beforehand which load host will actually down-line load the load host image. A **syslog** message informs you after

the load. (See the **syslog** description in the ULTRIX documentation for more information.)

After you issue the **trigger** command on one of the network's management hosts, the down-line load proceeds as follows (this system may also be one of your load hosts, but it is not required):

1. The management host sends a **mop remote console boot** message with the **trigger** option specified.
2. When the remote node receives this message, it multicasts a **mop request program** message.
3. The first load host that responds and the remote node, use additional **mop** messages to transfer the management host's software image into the remote node's memory. The remote node ignores other responders once the load is in progress.

1.8 Network Management Tasks

As manager of a DECnet–ULTRIX node, you have a number of key responsibilities, including:

- Configuring your node to ensure proper operation with other nodes in the network. (See Chapter 2.)
- Controlling access to the network. (See Chapter 3.)
- Monitoring Ethernet or DDCMP point-to-point operation. (See Chapter 4.)
- Testing Ethernet and DDCMP point-to-point hardware and software operation. (See Chapter 5.)

The following chapters tell you how to use **ncp** commands to perform each task. All DECnet–ULTRIX **ncp** commands discussed in these chapters are fully described in the *DECnet–ULTRIX NCP Command Reference* manual.

Chapter 2

Configuring Network Components

This chapter describes the network components and the procedures for configuring them with **ncp** commands. Each section describes how to specify the component and includes other procedures useful in configuring the component into the network.

All components are defined automatically during the installation of the DECnet–ULTRIX software. Use the **ncp list** command to display the parameter values for each component in the permanent database (see Chapter 4). You can define more nodes or modify existing ones in both the permanent and volatile databases.

Table 2-1 lists **ncp** commands commonly used to configure network components in the volatile and permanent databases.

Table 2–1: ncp Commands to View and Change Databases

ncp Function	Applicable Components and Devices	Volatile Database Command	Permanent Database Command
Display component data	circuit executor line logging[1] node object	**show**	**list**
Create/modify parameters for the specified component	circuit executor logging node object	**set**	**define**
Remove parameters for the specified component	circuit executor logging node object	**clear**	**purge**

[1]See Chapter 4 for information on how to specify event-logging procedures.

2.1 Configuring Nodes

Many functions the network manager performs require the identification of a specific node. During DECnet–ULTRIX installation, you define your local node name and address. You can also define node names and addresses for remote

nodes. The following sections describe node identification and the relevant node parameters for establishing an operational nodes database.

When configuring the network node database, you can use **ncp** to:

- Specify local, remote, and executor nodes by name and number
- Change the executor node identifier string
- Change the executor node state
- Reset a node's Ethernet address
- Enable proxy access on the executor node (see Chapter 3)
- Set up down-line load or up-line dump parameters

2.1.1 Specifying a Node

Specify node identification in one of the following forms:

- **Node address.** The numeric address of the node, consisting of an area number from 1 through 63, followed by a period and a number from 1 through 1,023. The second number represents the node number within the specified area. For example, a node address of 4.105 would represent node number 105 in area 4. If you are referencing a node within your area, you may omit the area number and the period separator.
- **Node name.** A unique alphanumeric character string containing 1 to 6 characters, including at least one alphabetic character.

Each node in the network must have a unique name and a unique address. Note that the node name is known only to the local node network software, while the node address is known networkwide by the routing function. Once you have specified both a node name and a node address, you can use either one whenever you need to specify a node identification in **ncp** commands.

2.1.2 Changing the Address of a Node

To change the name or address associated with a node, use the **set/define node** command.

EXAMPLE 1:

The following example changes node address 12 to node name BURGER:

```
ncp> set node 12 name burger RET
```

EXAMPLE 2:

To change the address of the executor node, execute the following command sequence:

```
set executor state off
purge node-address all
purge node node-name all
define executor address node-address
set executor state on
```

NOTE

1. If you are running Internet, Local Area Transport (LAT), or any other software that would be affected by a change in the Ethernet physical address of the executor, you must disable the software before issuing the **ncp set executor state on** and then enable it again.
2. If you have any LAT terminal lines on your system, you can use the **lcp** command to turn them off and then on again.
3. If you have Internet, you should also use the **arp -d** command to disassociate the executor with the old Ethernet physical address and to broadcast the new physical address to other Internet nodes.

2.1.3 Removing a Node

To remove a node name from the volatile database, use the **clear node** command. To remove a node from the permanent database, you must use the **purge node** command and specify the **all** keyword.

EXAMPLE 1:

The following command removes the node BOSTON from the volatile database:

```
ncp>clear node boston all RET
```

EXAMPLE 2:

The following command removes the node OHLONE from the permanent database:

```
ncp>purge node ohlone all RET
```

Some commands allow you to specify all known nodes instead of just one specific node. The **known nodes** keyword refers to all nodes that are defined in the database affected by the command you are using. For example, a **set known nodes** command affects all nodes currently defined in the volatile database but leaves node definitions in the permanent database unchanged.

2.1.4 Changing the Executor Node Identifier String

During DECnet–ULTRIX installation, you can provide a string of information that appears on the screen whenever you display executor node information.

To modify this string, use the **identification** parameter with the **set executor** or **define executor** command in the following format:

ncp set executor identification *id-string*

When you issue this **ncp** command at the shell prompt, you must follow these conventions: for the shell, enclose any string containing blanks or tabs in double quotation marks; for **ncp**, use single quotes.

EXAMPLE 1:

The following example shows the **ncp define executor identification** command executed from the shell:

```
% ncp define executor identification 'DECnet—ULTRIX BOSTON V4.0' RET
```

EXAMPLE 2:

This example shows the same command executed from within **ncp**:

```
ncp>define executor identification "DECnet—ULTRIX BOSTON V4.0" RET
```

EXAMPLE 3:

To quote a word within the identification string, enclose the word within two sets of double quotation marks. For example:

```
% ncp define executor identification 'DECnet—ULTRIX ""BOSTON"" V4.0' RET
```

2.1.5 Changing the Executor Node State

When your DECnet–ULTRIX system is installed, the executor node comes up in the **on** state and is available for use. If you do not want your system to come up with the executor node on, you must edit the **/etc/rc.local** file after installation and before rebooting to remove the **ncp set executor state on** command.

NOTE

If you have Local Area Transport (LAT) terminal lines on your system, you should not delete the **ncp set executor state on** command from the **/etc/rc.local** file unless you start TCP/IP with the **ifconfig** command. Either DECnet or TCP/IP must be running for LAT to function.

Once your system is running, you can use the following **set executor** command to change the state:

set executor state { **on** | **restricted** | **shut** | **off** }

where

on	specifies normal operation. Allows logical links to and from the node.
restricted	specifies limited operation. Allows no new inbound links.
shut	specifies orderly shutdown. Allows no new logical links to or from the node, but does not terminate existing links. After all links have disconnected, the executor goes into the **off** state, thereby shutting down the network at the local node.
off	specifies immediate shutdown. Immediately terminates all network activity without allowing active links to disconnect in an orderly manner. All active links are aborted and active tasks are notified of network shutdown.

2.1.6 Resetting a Node's Ethernet Address

Each DECnet node on the Ethernet has a unique node address that allows it to communicate with any other node on the same Ethernet. The Ethernet address on your Ethernet controller is reset by the DECnet software whenever the **ncp set executor state on** command is executed.

NOTE

This usually occurs during system installation. However, it can occur later if you redefine your node address.

2.1.7 Down-Line Loading or Up-Line Dumping a Node

One way to set up the parameters for down-line loading from a DECnet–ULTRIX node is to use the following command format:

ncp set node *node-id* **load file** *filename* RET

One way to set up the parameters for up-line dumping to a DECnet–ULTRIX node is to use the following command format:

ncp set node *node-id* **dump file** *filename* RET

For more information about using **ncp set node** or **define node** commands, see the *DECnet–ULTRIX NCP Command Reference* manual.

2.2 Configuring Network Objects

To configure a network object, use the following:

- Specify an object by name or number.
- Define an object file name.

2.2.1 Specifying an Object

All objects are identified by an object name and a number from 0 through 255, depending upon the type of object.

- **Zero objects** are user-defined processes for special-purpose applications. All objects assigned to object number 0 are identified by an object name (1 to 16 alphanumeric characters). You must use this object name when issuing a logical link connect request.

  An undefined object is a zero object that is not defined in your object list.

- **Nonzero objects** usually are DECnet-supplied processes, such as File Access Listener (**fal**) and Network Management Listener (**nml**), that provide a specific, cross-system network service. However, you can also supply user-written tasks for known network services. Each object is identified by an object number ranging from 1 through 255. DECnet-supplied objects that perform the same function on different systems are all assigned the same object number, even if their object names are different. For example, the DECnet–ULTRIX network terminal handler (**dterm**) and the DECnet–VAX terminal handler (REMACP) are both assigned object number 23. Numbers from 1 through 127 are reserved for Digital-supplied services; numbers from 128 through 255 are available for customer-supplied services.

2.2.2 Defining an Object File Name

Each DECnet-supplied object invokes an executable file that is defined for that object (use the **ncp show known objects** command to display file names defined for objects on your system). The DECnet-ULTRIX kit supplies a zero object named DEFAULT that does not have an executable file associated with it. You can use the **ncp set/define object** command to define a DEFAULT file name or you can leave the DEFAULT object file undefined, depending upon what action you wish to take when an undefined object name (that is, a zero object that is not defined in your object list) is received on a connect request:

- If DEFAULT has a defined file name, any zero object that is not defined in your object list executes the file defined for DEFAULT.
- If no file name is defined for DEFAULT and an undefined zero object is received on a connection request, the system software searches for the supplied object name according to the search path for the default user account associated with DEFAULT. If the software finds a file that matches the supplied object name, it executes that file. If it does not find a matching file name, it rejects the connection request.

To define the zero object, enter:

```
ncp> set object RET
```

or

```
ncp> define object RET
```

For information about the **ncp set object** or **define object** commands, see the *DECnet–ULTRIX NCP Command Reference* manual.

2.3 Configuring Lines

To configure a line do the following:

- Specify a line by device name and number.
- Change a line's state.

2.3.1 Specifying a Line

A line ID has the following format:

dev-c

where

dev is a DECnet–ULTRIX line device name. The following table contains the DECnet device names with the equivalent Internet names:

DECnet Device Names	Equivalent Internet Device Names
una	de
qna	xna
dmc	dmc
dmv	dmv
sva	se
bnt	ni

c is a number from 0 through 65,535 that designates the device's hardware controller.

NOTE

The **ncp** commands **list known lines** and **list known circuits** display only lines or circuits configured on the ULTRIX system and running on DECnet. If one of these commands does not display the circuit or line you are looking for, enter a **show** or **list** command for that entity.

The following command displays the circuit una-0:

```
ncp>list circuit una-0 RET
```

2.3.2 Changing a Line's State

When you install DECnet–ULTRIX, the line on which you choose to run DECnet turns on and is available for use when the system comes up.

DECnet and Internet can share the same Ethernet line. You can control DECnet's access to the Ethernet by setting the circuit's **state** with the **ncp set** or **define circuit** command.

EXAMPLE 1:

This command turns the DECnet network on:

```
ncp> set circuit una-0 state on RET
```

With an Ethernet line you can leave the line state on even when you are not using DECnet.

With a DDCMP point-to-point line you must switch between running DECnet and Internet. When you install DECnet–ULTRIX, DECnet is on. To switch to Internet, you must turn off DECnet, and then turn on Internet. The **ncp set exec state** command turns off DECnet, and **ifconfig(8)** turns on Internet.

EXAMPLE 2:

This command turns off DECnet:

```
% ncp set exec state off RET
% /etc/ifconfig dmv-0 boston 1.16 netmask 255.0 RET
```

To switch back to DECnet, turn off Internet and reset the DECnet executor state to **on**.

EXAMPLE 3:

This command turns off Internet and resets the DECnet executor state to **on**:

```
% /etc/ifconfig dmv-0 down RET
% ncp set exec state on RET
```

See the ULTRIX network management documentation for more information about turning Internet on and off.

For more information about configuring lines, see the *DECnet–ULTRIX NCP Command Reference* manual.

2.4 Configuring Circuits

To configure a circuit do the following:

- Specify a circuit by device name and number.
- Change a circuit's state.

2.4.1 Specifying a Circuit

To use **ncp** commands to modify or display circuit parameters, you must specify an individual circuit by using its circuit ID in the following form:

dev-c

where

dev	is one of the following DECnet–ULTRIX circuit names: una, qna, dmv, dmc, sva, or bnt.
c	is a number from 0 through 65,535 that designates the device's hardware controller.

2.4.2 Changing a Circuit's State

If you do not want the circuit turned on the next time your system comes up, you can issue an **ncp define circuit** *circuit-id* **state off** command to set the circuit off in the permanent database.

While your system is running, you can use the **set circuit** command to change the circuit state. Changing the circuit state controls DECnet access to the line without affecting the line state.

Chapter 3

Controlling Network Access

This chapter explains how to use access-control information for commands you want to execute on a remote node. It also tells you how to set up proxy for incoming connection requests.

3.1 Access Restrictions on Remote Nodes

Before you issue commands to be executed at a remote node, you may have to supply access-control information. Depending on the type of access restrictions set up by the system manager of the remote node, you can gain access to a remote node from a DECnet–ULTRIX node in the following ways:

- Append access-control information to the node identification in the **ncp** command string.
- Include access-control information in an alias definition for the node.
- Use proxy verification on the remote node.

For procedures on how to supply access-control information, see the *DECnet–ULTRIX Use* manual.

3.2 Appending Access-Control Information

If you append access-control information to the node identification on any **ncp** command line, the remote node verifies your log-in name and password. It does this by checking the log-in name and password against its password file (user authorization file). If the password file contains a matching entry, the remote node executes the command; if not, the node rejects the command.

Use the following format to append access-control information to the node name:

node-id **user** *login-name* [*password*]

You can also use the ULTRIX format:

node-id/login-name[*/password*]

where

node-id	is the name or address of the remote node.
login-name	is a string of up to 39 alphanumeric characters that identifies a user on the remote node. The log-in name for an ULTRIX user is the account name.

password is a string of up to 39 alphanumeric characters identifying the remote user's log-in name. The remote system uses this string to verify the identity of the user.

If you do not want the password to echo, type a question mark (?) in place of the *password* string.

If you are using the ULTRIX format, type a backslash and a question mark (?) in place of *password*. For example:

```
ncp>tell boston/jill/ \? set exec gateway access enable RET
```

After you press RET the system prompts you for your password and does not echo it as you type.

If you choose not to enter a password, press RET at the password prompt.

3.3 Defining an Alias

As a shortcut to typing the remote node identification and the required access-control information, you can specify an alias node name. An alias node name is an alphanumeric string of one or more characters that you type in place of a node identification and access-control information. You can define alias node names by creating a **.nodes** file in your home directory. Use the following format for entries in this file:

alias=node-id[/login-name[/password]]

NOTE

Do not use spaces or tab characters in any of the fields.

EXAMPLE 1:

This example shows three ways to add **.nodes** file entries:

```
b=boston
w=boston/root/xyzkoroijt
payroll=boston/hart/yxkowilk
```

EXAMPLE 2:

This example shows how the alias **w** represents the string **boston/root/xyzkoroijt** in a **tell** command string.

```
% ncp tell w set execution gateway access enable RET
```

CAUTION

To prevent unauthorized access to your passwords, set up the protections on the **.nodes** file so that only the owner can read the file or write to it.

3.4 Using Proxy Verification

Proxy verification lets you execute commands on a remote node without supplying access-control information. It is more secure than sending your password over the network. Although DECnet–ULTRIX software supports proxy verification, not all DECnet systems do. Contact the manager of any non-ULTRIX system to find out if it supports proxy verification.

Before you can use proxy verification, the system manager for the remote node must set up a proxy account for you. If you are going to have access to more than one proxy account from the same node and log-in name, indicate which proxy account should be the default.

To use your default account, enter the command without any access-control information.

EXAMPLE:

This example shows how to use the **tell** command to enter a command without access-control information.

```
ncp>tell navaho set exec gateway access enable RET
```

To use an account other than the default, append the account log-in name to the node identification.

EXAMPLE:

This example shows how to use the **tell** command to append the log-in name to the node identification.

```
ncp>tell boston/rudi set exec gateway access enable RET
```

3.5 Controlling Proxy Access on a Node

The following sections describe how to set up and manage a proxy file, control proxy access to your node, and control outbound proxy requests.

3.5.1 Setting Up a Proxy File

To set up a local proxy file, make an entry in the file **/etc/dnet_proxy** for each user to whom you want to permit proxy access. To set up more than one proxy account for a user, list the entry for the user's default account above any other entries for that user. Use the following format for the entries:

source-node-name::*source-login-name destination-login-name object* [*#comment*]

where

source-node-name	is the name of the remote node from which you are allowing proxy access on your node.
source-login-name	is the remote user for whom you are allowing proxy access.
destination-login-name	is the account on your node to which you are allowing proxy access.
object	is a list of objects, separated by commas, for which the proxy entry is valid. (Optional.)

#comment	is an alphanumeric string of one or more characters for notes about the proxy account.

You can use a wildcard character in place of the source node name, source log-in name, or destination log-in name to indicate that any entry is valid for that field.

The following examples show a proxy file and explain each of its four entries.

EXAMPLE 1:

This example shows a proxy file with four entries:

```
*::jones jones #Jane Jones
school::* newuser #account for new students
navaho::* guest fal
boston::* * #Accounts for node BOSTON users
```

EXAMPLE 2:

This entry means that remote user **jones** has access from any node to account **Jones** on your node.

```
*::jones jones #Jane Jones
```

EXAMPLE 3:

This entry indicates that any user on node SCHOOL has access to account **newuser** on your node.

```
school::* newuser #account for new students
```

EXAMPLE 4:

This entry indicates that any user on NAVAHO can access object **fal** through the account **guest**.

```
navaho::* guest fal
```

EXAMPLE 5:

This entry indicates that any user on node BOSTON can access an account with the same name on your node.

```
boston::*     * #Accounts for node BOSTON users
```

CAUTION

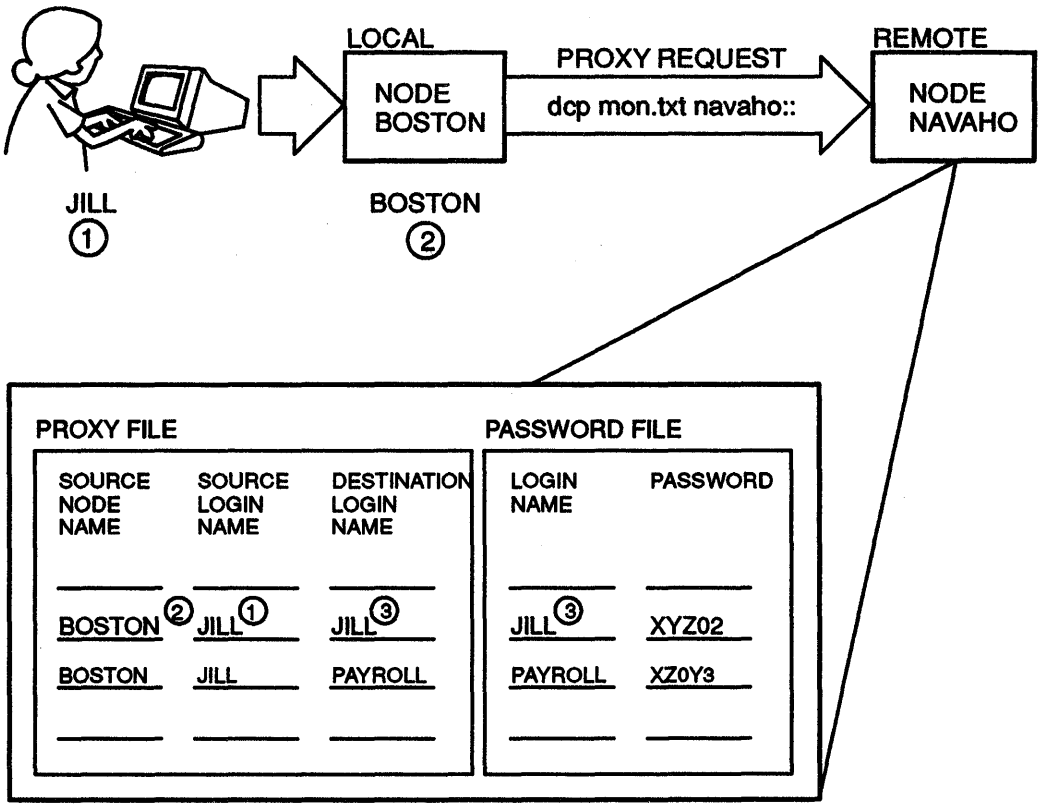
For added security, protect the proxy file so that only the superuser (privileged) can access it.

If a remote user's connection request does not contain access-control information, the following conditions must be met for proxy to be approved:

- The proxy file must contain a source node and log-in name combination that matches the remote user's source node and log-in name. For example, if a remote user is logged in to node BOSTON as user **jill** when she executes an **ncp** command, the proxy file must contain a source node name BOSTON with the source log-in name **jill**.
- The system password file must contain a user name that matches the proxy file entry's destination log-in name.

Figure 3-1 shows a remote user, a proxy file, and a password file. Lines connect the information that must match for the executor node to accept the remote user's request.

Figure 3–1: Proxy Request Without Access-Control Information

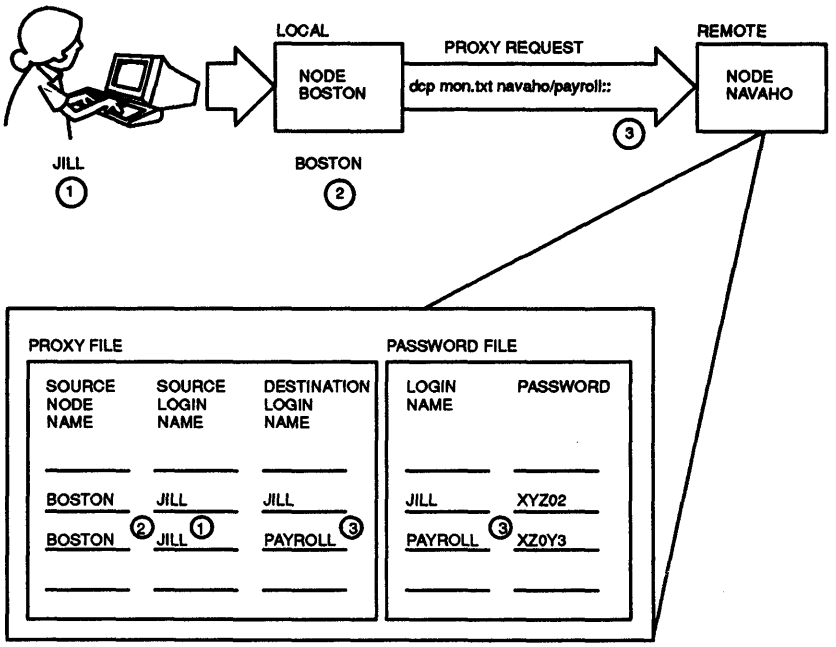


If a remote user appends a user name to the node identification, the following must hold true for proxy to be approved:

- The proxy file must contain an entry in which the source node and log-in name combination matches the remote user's source node and source log-in name.
- This proxy file entry must also contain a destination log-in name that matches the user name that the user appended to the node identification.
- The system password file must contain a user name that matches the proxy file's destination log-in name.

Figure 3–2 shows the required matches:

Figure 3–2: Proxy Request With Access-Control Information



LKG–3777–90I

If any one of these requirements is not met or **incoming proxy** is disabled, the system can approve access only if the user name appended to the node identification is listed in the system password file with a null password.

3.5.2 Enabling or Disabling Incoming Proxy

The executor parameter **incoming proxy** lets you control connect requests to your node. If proxy is disabled, the system will reject an incoming connection request. This parameter is enabled by default. To turn off proxy access to your node, set this parameter to **disable**.

EXAMPLE:

This example shows how to use the **set executor** command to turn off proxy access:

```
ncp> set executor incoming proxy disable RET
```

The default for the **incoming proxy** parameter is **enable**.

3.5.3 Enabling or Disabling Outgoing Proxy

You can prevent outbound connection requests from leaving your node by setting the executor's **outgoing proxy** parameter to **disable**.

EXAMPLE:

This example shows how to use the **set executor** command to disable outbound proxy requests.

```
ncp> set executor outgoing proxy disable RET
```

The default for this parameter is **enable**. For more information, see the **set executor** or **define executor** command in the *DECnet–ULTRIX NCP Command Reference* manual.

3.6 Using a Default User Account

You can define a default user account, and then specify the default user for the requested object (service) in the object database. Unless you redefine the default user, all DECnet objects use "guest" as a default user name.

For example, if a user tries to copy a file to your node and **incoming proxy** is disabled, the DECnet object spawner checks to see whether the object **fal** has a default user. When the spawner finds that guest is the default user, it looks for guest in the password file. If the spawner finds guest in the password file, it verifies access and **fal** copies the file into the guest account.

Chapter 4

Monitoring Network Activity

DECnet provides several means of monitoring network activity from a DECnet node. For example, you can:

- Display information about network components by using the **ncp show** or **list** commands.
- Use **ncp** to request the Event Logger (**evl**) to log network events.
- Measure network performance by evaluating the DECnet counters for circuits, lines, and nodes.
- Monitor access to your local node by logging certain object activities to a log file.

All of these facilities are described in this chapter. See the *DECnet–ULTRIX NCP Command Reference* manual for details on the **ncp** commands.

4.1 Using ncp Display Commands

The **ncp show** and **list** commands display information about network components. The **list** command displays information from the permanent database, while the **show** command displays component information from the volatile database.

The components for which you can display information include:

- Network components:
  - Circuits
  - Lines
  - Nodes
  - Objects
- Event Logger components:
  - Console
  - File
  - Monitor program

You can display information for a specific component or for all known components of the specified type. For example, if you want to see which nodes are currently reachable, you can issue a **show known nodes** command. You can also display a subcategory of **known** for nodes by specifying **show active nodes**, which displays information about known nodes that are currently turned on.

Depending on the component you specify in a **show** command, you can select from the following displays:

characteristics	displays static information about the component, such as the parameters defined for that component. This information is kept in either the volatile or permanent database.
counters	provides counter information for circuits, lines, and nodes. (See Section 4.3 for a discussion of counters and a sample display.)
events	displays information about events currently being logged. Valid for **show** or **list logging** only.
status	shows information that usually reflects network activity for the running network. Depending on the component, this information can include the local node and its operational state, reachable and unreachable nodes, and circuits and lines and their operational states.
summary	includes only the most useful information, which is usually an abbreviated list of information provided for both the **characteristics** and **status** display types. (Default.)

The following examples demonstrate some **ncp show** commands and their resulting displays.

EXAMPLE 1:

To display line characteristics, enter:

```
ncp> show line una-0 characteristics RET

Line Volatile Characteristics as of Wed Jun 19 16:38:06 EDT 1990

Line  =  UNA-0

Controller              = Normal

Protocol                = Ethernet

Hardware address        = aa-00-03-00-00-99
```

EXAMPLE 2:

To display characteristics of a specific object, enter:

```
ncp> show object nml char RET

Object Volatile Characteristics as of Thu Jun 14 10:07:11 EST 1990

Object = nml

Number                          = 19
File                            = /etc/nml
Default User                    = guest
Type                            = Sequenced Packet
Accept                          = Deferred
```

4.2 Using Event Logging

After the DECnet–ULTRIX software is installed, **evl** begins recording network events, such as changes in node state, loopback test messages, and line and node counter activity.

By default, all events for all components known to the local node are logged at the executor console. However, you can use **ncp set logging** or **define logging** commands to specify the event classes or types you want to log. You can also inhibit or enable logging to the console, a file, or user program.

The following sections describe how to customize the Event Logger.

4.2.1 Displaying Event Logger Status

Before you can begin customizing the Event Logger, view the status of the existing components.

EXAMPLE:

To display all existing logger components, enter:

```
ncp> show known logging RET

Known Logging Volatile Summary as of Thu Jun 14 10:10:51 EST 1990

Logging = console

State                          = On
Name                           = /dev/console
Sink node                      = 7.3 (ATLANT), events =
                                   0.0-6
                                   2.0-1
                                   3.2
                                   4.3-10,18

Logging = file

State                          = Off
Name                           = /usr/adm/eventlog

Logging = monitor

State                          = On
Name                           = /etc/evl
```

4.2.2 Specifying an Event Class and Type

Events are defined by class and type. You can specify class and type of events to be logged by using the following event list format with the **events** parameter in a **set** or **define logging** command:

class.type[,*type*,*type*,...,*type*]

where

class	identifies the DNA layer or system-specific resource to which the event pertains.
type	identifies the particular event within the event class. The *type* variable can be a single number or a range of numbers.

When providing an event list for the **events** parameter, you can specify only one class, but you can specify multiple event types within a class. For example, you can specify a single event type, a range of types, a combination of these, or a wildcard character. The following sample event list illustrates the different formats:

Event List	Meaning
2.0	Specifies event class 2, type 0.
4.15-18	Specifies event class 4, types 15 through 18.
4.3,8-10,18	Specifies event class 4, types 3, 8 through 10, and 18. Note that types must be specified in ascending order.
0	Specifies all events in class 0.

See the *DECnet–ULTRIX NCP Command Reference* manual for a list of all events (by class and type) that DECnet–ULTRIX can log. To determine the events logged by a remote node, see the DECnet documentation for the remote system. To specify logging of all network event classes and types, use the **known events** parameter.

4.2.3 Event Sources

Event sources are qualifiers for events. If no source is specified, logging events for all sources are affected by the command. When you specify a source for events, only events generated by that source are affected. Sources you can specify for DECnet–ULTRIX events are **circuit**, **line**, or **node**.

EXAMPLE:

To monitor network activity over line una-0, connected to the local node, use the following command:

```
ncp> set logging console known events line una-0 RET
```

This command causes all events that pertain to line una-0 to be logged at the console by the Event Logger.

4.2.3.1 Specifying Logging Component Names

When you initially specify logging components to which event data is to be logged, you can identify them by using the **name** parameter in the **set logging** command. Use the **clear logging name** command to clear the name of the logging component and cause events to be sent to the default device for that logging component. The default logging names are as follows:

- For the console logging component: **/dev/console**
- For the file logging component: **/usr/adm/eventlog**
- For the monitor logging component: **evl**

4.2.3.2 Logging Sinks

You can use a sink qualifier with any of the logging components. Use the **sink** parameter to indicate the location of the logging component. The sink can be the executor node or a remote node.

EXAMPLE:

This command routes all events to the console logging component on node NAVAHO.

```
ncp> set logging console known events sink node navaho RET
```

If you do not specify a sink qualifier, the local node is the default component location. If the sink node is unreachable when the event occurs, the event information is discarded.

4.2.3.3 Event-Logging Component States

You can inhibit or enable logging to the console, a file, or user program by setting the **state** parameter of that component to **on** or **off**. The **state** parameter causes all events destined for the logging component to be discarded. The **on** state enables logging on the specified logging component.

EXAMPLE:

This command disables logging to the system operator's terminal on the local node.

```
ncp> set logging console state off RET
```

NOTE

This does not affect logging to any other logging component whose state may be **on**.

4.2.4 Monitoring Local and Remote Nodes

You can use event-logging commands to monitor local and remote nodes in your network.

EXAMPLES:

The sequence of **ncp** commands, shown in the following examples, sets up event logging for a local node called TOM and for two remote nodes called DICK and HARRY.

EXAMPLE 1:

In response to this command, TOM's system console displays all known events that occur locally:

```
ncp> set logging console known events state on RET
```

EXAMPLE 2:

In response to the following four commands, remote nodes DICK and HARRY each log all known events locally at the system console and remotely at node TOM's system console.

```
ncp> set logging console known events state on RET
ncp> tell dick set logging console known events state on RET
ncp> tell dick set logging console sink node tom known events RET
ncp> tell harry set logging console known events state on RET
ncp> tell harry set logging console sink node tom known events RET
```

4.3 Reading Counters

DECnet–ULTRIX software maintains certain statistics, called counters, for circuits, lines, and nodes (including the executor). All counters are listed and briefly described in the *DECnet–ULTRIX NCP Command Reference* manual.

Counters are maintained on the node presently designated as the executor and can include such information as the number of data packets sent, received, or lost over a line; the number of connect messages sent or received; system buffer allocation failures; and routing packet information. Counter statistics are useful alone or when read in conjunction with logging information to measure and evaluate the performance and throughput of your network configuration.

You can display counters and periodically reset them to zero using the **show** and **zero** commands.

4.3.1 Displaying Counters

You can use the **ncp show** command described in the following examples to display counters for circuits, lines, and nodes.

EXAMPLE:

This example shows a sample command and the resulting display:

```
ncp> show line una-0 counters RET
Line Volatile Counters as of Wed Jun 19 16:40:22 EDT 1990
Line  =  UNA-0
  12770 Seconds since last zeroed
5703244 Bytes received
3695497 Bytes sent
2498504 Multicast bytes received
  87029 Data blocks received
  65294 Data blocks sent
  23688 Multicast blocks received
   1477 Blocks sent, initially deferred
     91 Blocks sent, single collision
     83 Blocks sent, multiple collisions
      0 Send failure
      0 Collision detect check failure
      3 Receive failure, including:
          Framing error
          Frame too long
   1996 Unrecognized frame destination
      0 Data overrun
      0 System buffer unavailable
      0 User buffer unavailable
```

The Ethernet header size is incorrectly included in the count for Bytes received and Multicast Bytes received on DESVAs and DEQNAs.

Some counters can be qualified by information that indicates the condition(s) that contributed to an error.

Some network logging events relate to the network counters; for example, event 0.5 logs node counter values before they are zeroed. See the *DECnet–ULTRIX NCP Command Reference* manual for a complete description of events that can be logged.

4.3.2 Zeroing Counters

When the network is running, you can use the **ncp zero** command to reset any of the network counters to zero. It is wise to zero counters periodically so that they do not exceed (overflow) their maximum count. Counters that have overflowed display a "greater than" sign (>) in front of their maximum count (for example, >65534 means that the maximum count of 65,534 has been exceeded).

Each component has a special counter that indicates the number of seconds that have elapsed since the counters for that component were last zeroed. The software increments this counter every second and zeros it when other counters for the component are zeroed.

4.4 Monitoring Access with Object Log Files

DECnet–ULTRIX logs specific information that can be used by the network manager to monitor user access of the local system. Four programs use the ULTRIX **syslog** subroutine to log such information: the DECnet object spawner, the File Access Listener (**fal**) and the remote terminal access server programs (**dlogin** and **dterm**). For more information on **syslog**, see **syslog(3)** in the ULTRIX reference documentation.

The user-access information is logged to file **/usr/spool/mqueue/syslog** by default. Table 4-1 lists the DECnet programs that log this information and notes the priority level and type of information logged for each.

Table 4–1: DECnet–ULTRIX Log Files

Program	Priority Level	Type of Information Logged
dnet_spawner	LOG_DEBUG(9)	All connects to and exits from DECnet objects. Also logs connect requests that are rejected by the DECnet spawner (for example, requests with bad access-control information).
fal	LOG_INFO(8)	All attempts at file access.
dlogin	LOG_INFO(8)	All remote terminal connects and disconnects. Supports DECnet heterogeneous command terminal specification (CTERM) for DECnet–ULTRIX and VMS V4.0 and later.
dterm	LOG_INFO(8)	All remote terminal connects and disconnects. Uses TOPS-20 homogeneous command terminal protocol.

You can set up the **syslog** configuration file (**/etc/syslog.conf**) to have this information logged to any files that you want. You can filter out logging of priority 9 data by setting the file priority to 8, or you can bypass logging of all of these programs to a file by setting the file priority to 7 or lower.

Chapter 5

Testing the Network

This chapter describes loopback and DECnet Test Sender /DECnet Test Receiver (**dts/dtr**) tests and explains how to use them.

5.1 Loopback and dts/dtr Tests

Several kinds of tests help you determine whether the network is operating properly. Among them are loopback and **dts/dtr** tests, which you can run on your nodes.

- **Loopback Tests.** Loopback tests (node level and circuit level) let you exercise network software and hardware by first sending data through various network components and then returning that data to its source for data comparison. After you have started DECnet–ULTRIX software, you can use **ncp loop** commands to test network performance.
- **dts/dtr Tests.** The **dtr** program functions as a slave to **dts** and must exist as defined object 63 at the remote node. The **dts** program initiates each test by issuing a connection request to **dtr**. Parameter information pertinent to the type of test requested is passed by **dts** to **dtr** in the optional data of the connection request. The **dts** user interface enables the user to issue commands with options to customize the test to be performed. Parameters are available to regulate such variables as message length, test duration, and type of data used.

5.2 Node-Level Loopback Tests

Node-level loopback tests check the logical link capabilities of a node by exchanging test data between DECnet tasks in two different nodes or between DECnet tasks in the same node. Two types of node-level tests allow you to test different layers of DECnet–ULTRIX software:

- **Local-to-local loopback tests** verify local-node network operation.
- **Local-to-remote loopback tests** verify network operation between the local node and a remote node.

Use the node-level tests first. Then, if further testing is desired, use the circuit-level tests.

To perform these loopback tests, use the **ncp loop node** command. The **loop node** operation uses the two cooperating tasks Looper and Loopback Mirror (**mir**). When you issue the **loop node** command, you can specify the following information:

- The **with** parameter, which specifies the type of binary information used to perform the test. The possible values are **ones**, **zeros**, and **mixed**. The value **mixed**, a combination of ones and zeros, is the default.
- The **count** parameter, which specifies the number of data blocks to be sent during the test. It is a number from 1 (default) through 65,535.
- The **length** parameter, which specifies the length in bytes of each block to be looped. This value must be a decimal integer from 1 through *n*, where *n* must be less than the smaller of either the local looper buffer size or the remote mirror buffer size. The default is 40 bytes.

If a test completes successfully, **ncp** prompts you for the next command. If a looped message returns with an error, the test stops and **ncp** prints a message specifying the reason for the failure. If a connection was attempted, **ncp** provides a count of the messages that were not returned.

EXAMPLE:

In the following test, a network manager attempts to send a message 10 blocks long to node BOSTON. The result is that the message is not looped because node BOSTON is unreachable.

```
ncp>loop node boston count 10 RET
ncp - listener response: Mirror connect failed, Node unreachable
Unlooped count = 10
ncp>
```

See the *DECnet–ULTRIX NCP Command Reference* manual for a complete list of error messages.

5.2.1 Local-to-Local Loopback Test

The local-to-local loopback test evaluates the local node's DECnet software using an internal logical link path; no physical device is used.

EXAMPLE:

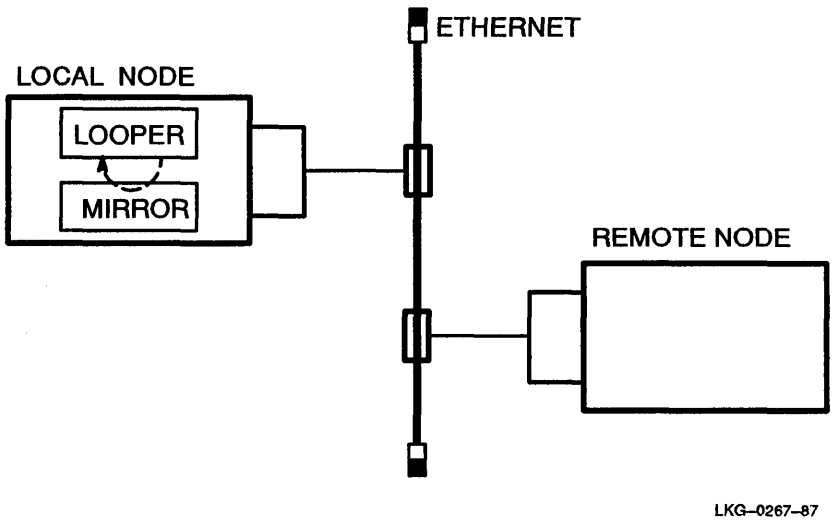
For this test, you simply issue the **ncp loop executor** command:

```
ncp>loop executor count 10 RET
```

The local-to-local loopback test verifies operation of the local Network Application layer, Session Control layer, End Communications layer, and part of the Routing layer. A failure of this test indicates a problem with the local node software, such as the network being turned off or access control to the mirror not being properly established. If the local-to-local loopback test succeeds, you should perform a local-to-remote loopback test. If the local-to-remote loopback test succeeds and the local-to-local test fails, try the circuit-level tests to determine if the hardware is at fault.

Figure 5-1 illustrates a local loopback test.

Figure 5–1: Local-to-Local Loopback Test



5.2.2 Local-to-Remote Loopback Test

This test verifies operation of all levels of network software on the local and remote nodes you are testing. When you use this command, you must identify the node to which you want to loop test messages. This node must be reachable over circuits that are in the **on** state.

Figure 5-2 illustrates a local-to-remote loopback test.

Before doing this test, use the **ncp show circuit summary** command to see whether the test circuit's state is **on**. If not, issue an **ncp set circuit state on** command for that circuit. Then use the **ncp loop node** command to initiate the loopback test.

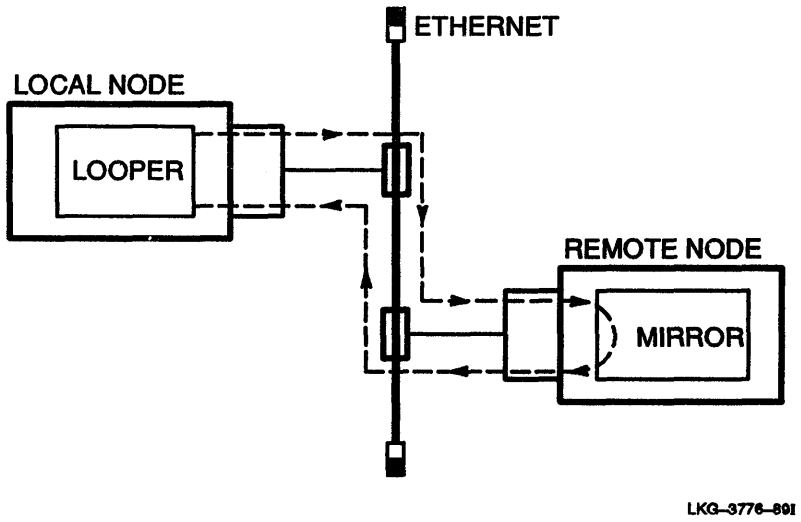
EXAMPLE:

The following command tests DECnet software on both the local node and on remote node TOLEDO:

```
ncp> loop node toledo count 10 RET
```

If the previous local-to-local tests were successful and this test fails, a problem exists with either the remote node or the DECnet circuit.

Figure 5–2: Local-to-Remote Loopback Test



5.3 Circuit-Level Loopback Tests

Circuit-level loopback tests check a DECnet circuit by looping test data between DECnet tasks in two different nodes or between DECnet tasks in the local node. These tests use a low-level data link interface rather than the logical links used by the node-level tests. There are two types of circuit-level tests:

- **Software loopback tests** loop the data through an adjacent node on the circuit to determine whether the circuit is operational up to the adjacent node's circuit unit and controller.
- **Controller loopback tests** loop the data through the device on the circuit in loopback mode to determine whether the controller works properly.

To perform these loopback tests, use the **ncp loop circuit** command.

When you run DECnet software on a diskless client, do not set the controller loopback mode with the **ncp** command **set line** *dev*-n **controller loopback**. Setting the controller to loopback mode causes the client to lose contact with its server and, as a result, to lose access to its file system.

The **ncp loop circuit** commands may not work when you issue them from a VMS node to an ULTRIX node on a DDCMP point-to-point line or an Ethernet cluster; they will work from an ULTRIX node to a VMS node.

When you issue the **loop circuit** command, you can specify the following information:

- The **with** parameter, which specifies the type of binary information used for the test. The possible values are **ones**, **zeros**, and **mixed**. The value **mixed**, a combination of ones and zeros, is the default.
- The **count** parameter, which specifies the number of data blocks to be sent during the test. It is a number from 1 (default) through 65,535.

- The **length** parameter, which specifies the length (in bytes) of each block to be looped. This value must be a decimal integer in the range of 1 through *n*, where *n* must be less than the smaller of either the local looper buffer size or the remote mirror buffer size. On the Ethernet, the allowable length is from 1 byte to the maximum length of the data pattern, which varies according to the level of assistance. The default is 40 bytes.

Level of Assistance	Maximum Length
No assistance	1486 bytes
Transmit or receive assistance	1478 bytes
Full assistance	1470 bytes

If a test completes successfully, **ncp** prompts you for the next command. If a looped message returns with an error, the test stops and **ncp** prints a message specifying the reason for the failure. If a connection was attempted, **ncp** provides a count of the messages that were not returned.

EXAMPLE:

This test attempts to send 10 messages. The first four messages are sent successfully, and an error occurs on the fifth, causing the test to halt.

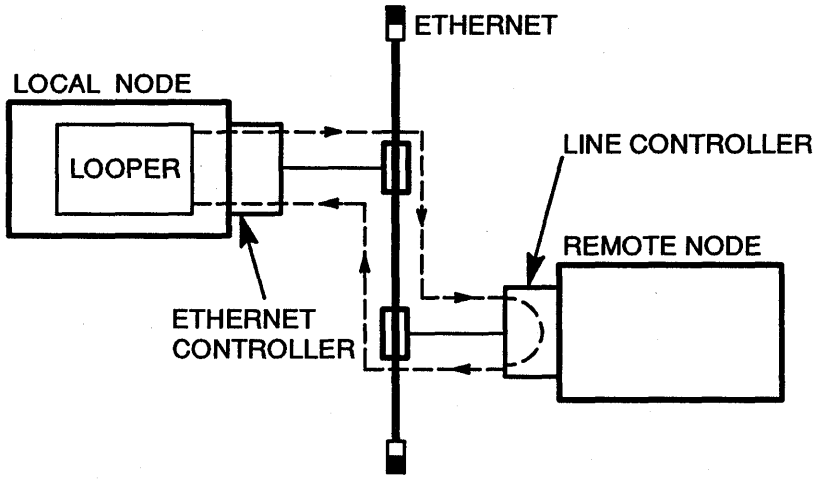
```
ncp> loop circuit una-0 count 10 RET
ncp - listener response: Line communication error
Unlooped count = 6
ncp>
```

5.3.1 Software Loopback Test

This test verifies that the circuit is operational up to the unit and controller on the adjacent node. This type of test uses DECnet–ULTRIX software to loop data through the circuit-to-circuit service software in the adjacent node and back to the local node. You can specify optional parameters for assistance in testing a remote node. Figure 5-3 illustrates a software loopback test.

Before doing this test, use the **ncp show line characteristics** command to see whether the line controller is in **normal** mode; if it is not, issue an **ncp set line controller normal** command for the line. Then issue an **ncp loop circuit** command to test the circuit between your node and an adjacent node. If this test fails, try a controller loopback test to see whether the controller is functional.

Figure 5–3: Circuit-Level Software Loopback Test



5.3.1.1 Software Loopback Testing Over Ethernet Devices

There are two ways of performing the software loopback test on the Ethernet:

- Loop to any random node on the Ethernet.
- Loop to a specific node on the Ethernet.

Before performing either test, issue an **ncp show circuit summary** command to see whether the circuit is in the **on** state; if it is not, issue an **ncp set circuit state on** command for that circuit.

Random Node Loopback Testing: You can send a test message to all nodes on the Ethernet by way of a multicast message.

EXAMPLE 1:

This command sends a test message to all nodes on the Ethernet. If the **count** parameter is greater than 1, the first node to respond to the multicast message will loop the remaining test messages.

```
ncp> loop circuit una-0 count 50 RET
```

EXAMPLE 2:

A return message is displayed that indicates the responding node's physical address:

```
Loop succeeded
Physical Address = AA-00-04-00-23-04
```

Specific Node Loopback Testing: You can specify a remote node on the circuit that you want to test by using either its node name or physical address.

NOTE

Nodes on Ethernet circuits are identified by unique Ethernet addresses. If the node is running DECnet, this physical address is the address that DECnet has created using the DECnet node address. If the node is not running DECnet, the physical address is the default hardware address of the node.

To perform the test using a specific remote node, specify the **physical address** parameter along with its value.

EXAMPLE:

This command loops messages through the local device una-0 to the remote node having physical address AA-00-03-00-FF-08:

```
ncp>loop circuit una-0 physical address aa-00-03-00-ff-08 RET
```

5.3.1.2 Ethernet Loopback Assistance

DECnet supports the use of an assistant node to aid you in interrogating a remote node. To use this assistant feature, specify either the **assistant physical address** parameter or the **assistant node** parameter as an additional parameter to the **ncp loop circuit** command.

You can choose one of three different kinds of help from the assistant, depending upon the **help** parameter value that you specify:

- **help full**—The assistant aids in both transmitting messages to and receiving messages from a remote node.
- **help transmit**—The assistant aids in transmitting loop messages to a remote node.
- **help receive**—The assistant aids in receiving loop messages from a remote node.

If you specify either the **assistant physical address** or **assistant node** parameter and do not specify the **help** parameter, you will receive **full** assistance by default. The three types of assistance are shown in Figures 5-4, 5-5, and 5-6.

Figure 5–4: Loopback Test Using Full Assistance

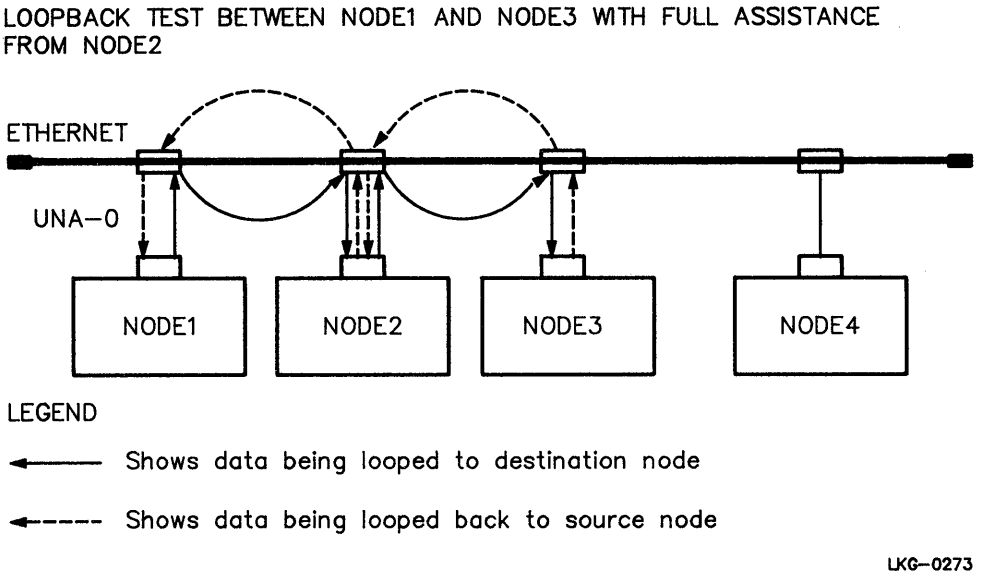


Figure 5–5: Loopback Test Using Transmit Assistance

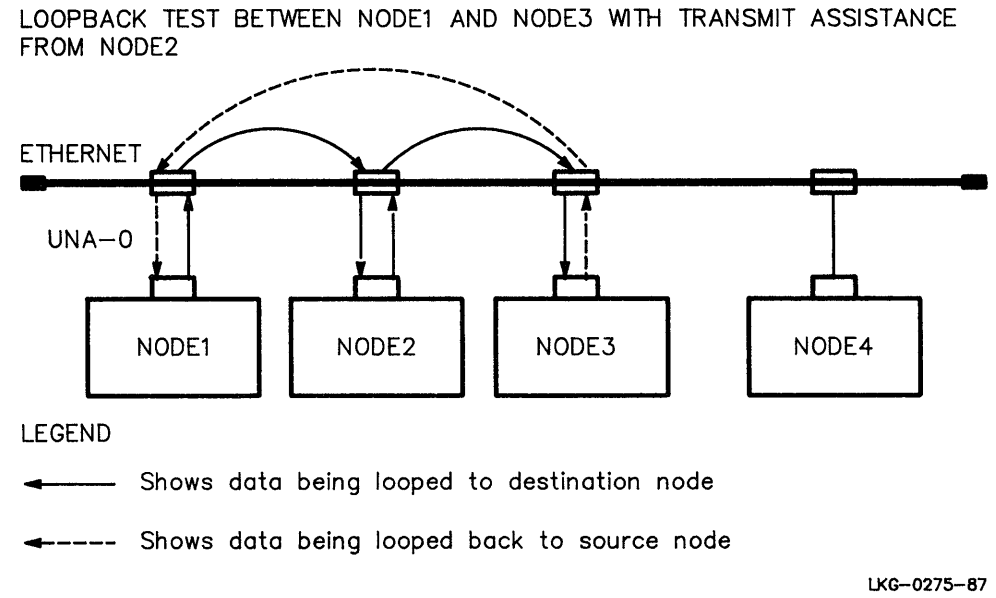
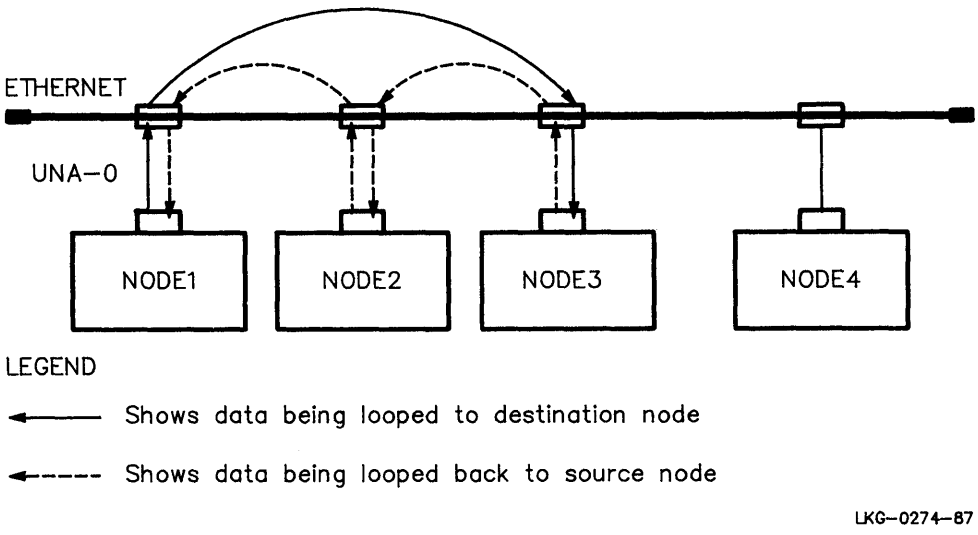


Figure 5–6: Loopback Test Using Receive Assistance



There are various reasons why you might choose one form of assistance over another. For example, if the target node to which you want to transmit a message is not receiving messages from your node, you can request assistance in transmitting messages to it. Similarly, if your node is able to transmit messages to the target node but not able to receive messages from it, you can send a message directly to the target node and request the assistant's aid in receiving a message back. If you encounter difficulties in both sending and receiving messages, you can request the assistant's aid in transmitting and receiving messages.

The following examples illustrate the use of the **assistant physical address** and **assistant node** parameters:

EXAMPLE 1:

In this command, you request the node described by Ethernet physical address AA-00-04-00-15-04 to assist you in testing the node described by Ethernet physical address AA-00-04-00-18-04. Since **assistant physical address** is specified without the **help** parameter, **full** assistance is given.

```
ncp> loop circuit una-0 physical address aa-00-04-00-18-04
 assistant physical address aa-00-04-00-15-04 RET
```

EXAMPLE 2:

In this command, you request node THRUSH to assist in testing node LOON by transmitting the loopback data to node LOON.

```
ncp> loop circuit una-0 node loon assistant node thrush help transmit RET
```

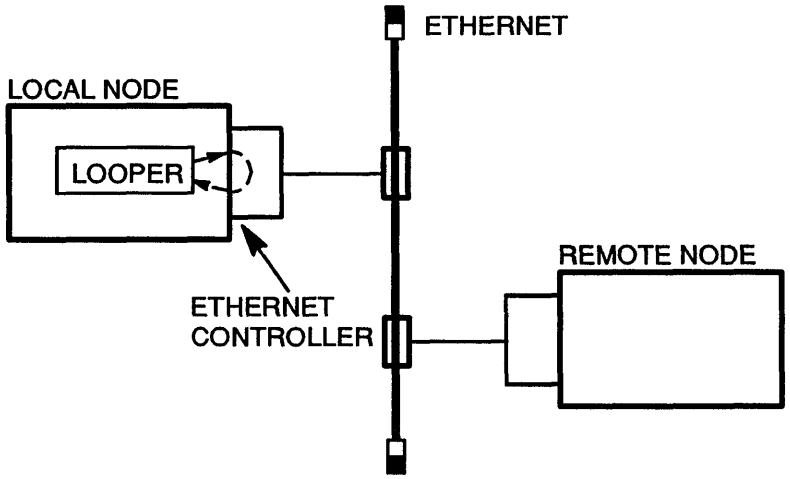
5.3.2 Controller Loopback Test

The controller loopback test verifies whether the circuit to the controller and the controller itself are functional. Figure 5-7 illustrates a controller loopback test.

NOTE

You must have superuser privileges for the controller loopback test.

Figure 5–7: Circuit-Level Controller Loopback Test



Before you begin this test, issue an **ncp show circuit summary** command to see whether the circuit state is **on**. If the circuit is off, use the **ncp set circuit state** command to turn it on.

To begin the controller loopback test, set the controller to **loopback** mode by using the **set line** command; then issue the **ncp loop circuit** command.

EXAMPLE:

This set of commands tests the circuit up to the controller for physical line una-0 connected to the local node by circuit una-0.

```
ncp> set line una-0 controller loopback RET
ncp> loop circuit una-0 count 10 length 32 RET
Loop succeeded
ncp> set line una-0 controller normal RET
```

If this test succeeds and the software loopback test fails, perform a circuit loopback test to see whether the device and device cable are functional.

5.4 The dts/dtr Tests

There are four basic tests provided by **dts/dtr**:

- Connect test
- Data test
- Disconnect test
- Interrupt test

Each test is divided into a set of subtests. The following sections describe the tests and subtests.

5.4.1 Connect Tests

Connect tests verify that the receiving node (**dtr** node) can process connection requests and return acceptance and rejection messages with or without optional user data. You can perform the following connect tests:

- **Connect reject without user data.** The **dts** node sends a connection request to the **dtr** node. The **dtr** node returns a rejection message that does not contain optional data.
- **Connect accept without user data.** The **dts** node sends a connection request to the **dtr** node. The **dtr** node returns an acceptance message that does not contain optional data.
- **Connect reject with 16 bytes of standard user data.** The **dts** node sends a connection request to the **dtr** node. The **dtr** node returns a rejection message that contains 16 bytes of optional data.
- **Connect accept with 16 bytes of standard user data.** The **dts** node sends a connection request to the **dtr** node. The **dtr** node returns an acceptance message that contains 16 bytes of optional data.
- **Connect reject with received user data used as reject user data.** The **dts** node sends a connection request that contains optional data to the **dtr** node. The **dtr** node returns a rejection message that contains the same optional data.
- **Connect accept with received user data used as accept user data.** The **dts** node sends a connection request that contains optional data to the **dtr** node. The **dtr** node returns an acceptance message that contains the same optional data.

5.4.2 Data Tests

Data tests provide a full range of tests from very simple data sink operations through data integrity checking. You can perform the following data tests:

- **Sink test.** The **dtr** program ignores all data received. No sequence or content validation is performed.
- **Sequence test.** Data messages transmitted by **dts** to **dtr** include a 4-byte sequence number. If a message is received out of sequence, **dtr** aborts the logical link and the test.

- **Pattern test.** Data messages transmitted to **dtr** have both a sequence number and a standard data pattern. If neither the sequence number nor the received data matches the expected data, **dtr** aborts the logical link and the test.
- **Echo test.** Data messages received by **dtr** are transmitted back to **dts**, checks the sequence number and data of the returned messages. If either is incorrect, **dts** aborts the link and the test.

5.4.3 Disconnect Tests

Disconnect tests verify that the receiving node (**dtr** node) can send out disconnect messages with or without optional user data. You can perform the following disconnect tests:

- **Disconnect without data.** The **dtr** node sends a disconnect message that does not contain user data to the **dts** node.
- **Abort without user data.** The **dtr** node sends an abort message that does not contain user data to the **dts** node.
- **Disconnect with 16 bytes of standard user data.** The **dtr** node sends a disconnect message that contains 16 bytes of optional data to the **dts** node.
- **Abort with 16 bytes of standard user data.** The **dtr** node sends an abort message that contains 16 bytes of optional data to the **dts** node.
- **Disconnect with received connect user data used as disconnect user data.** The **dts** node sends a message containing optional user data to the **dtr** node. The **dtr** node returns a disconnect message containing the same optional data to the **dts** node.
- **Abort with received connect user data used as abort user data.** The **dts** node sends a message containing optional user data to the **dtr** node. The **dtr** node returns an abort message containing the same optional data to the **dts** node.

5.4.4 Interrupt Tests

Interrupt tests provide a full range of test capabilities from very simple data sink operations through data integrity checking. Interrupt tests that the user can perform are as follows:

- **Sink test.** The **dtr** program ignores all interrupt data received. No sequence or content validation is performed.
- **Sequence test.** Interrupt messages transmitted by **dts** to **dtr** contain a 4-byte sequence number. If a message is received out of sequence, **dtr** aborts the logical link and the test.
- **Pattern test.** Interrupt messages transmitted to **dtr** have both a sequence number and a standard data pattern. If neither the sequence number nor the data pattern received matches the expected data, **dtr** aborts the logical link and the test.
- **Echo test.** Interrupt messages received by **dtr** are transmitted back to **dts**, which checks the sequence number and data of the returned messages. If either is incorrect, **dts** aborts the link and the test.

5.5 Performing dts/dtr Tests

The following procedure shows how to set up and run **dts/dtr** tests.

1. Be sure that the DECnet communications line is in the **on** state.
2. Enter the following command:

   % dts

   The system responds with a message like the following and a prompt:

   ```
   DTS initiated on Mon Feb 26 13:06:22 1990
   (DECnet-ULTRIX)

   DTS>
   ```

3. Enter **dts** commands at the command prompt.
4. To end testing, type **exit** in response to the **dts** prompt. The **dts** program prints a termination message on your screen when it exits, and the ULTRIX prompt reappears.

You can also enter **dts** commands with a **dts** command file.

EXAMPLE:

This command instructs **dts** to process the commands contained in the file **dtsshell** and to redirect the output to logging file **dts.log**.

```
% dts <dtsshell >dts.log
```

The exit status returned by **dts** commands is useful in a shell script.

5.5.1 Command Syntax for dts

Use the following format when entering **dts** commands:

dts>*test*[*qualifiers*][*test-specific-qualifiers*]

where

test specifies the type of test, which must be one of the following:

connect	connect test
disconnect	disconnect test
data	data test
interrupt	interrupt test

qualifiers specifies any number of the following optional qualifiers. Once specified, these qualifiers remain in effect for all applicable tests until you change them or exit from **dts**. Each qualifier must be preceded by a slash (/).

/nodename=*node-id* specifies the name or address of the DECnet node on which you want **dtr** to run. The default is the local node. If you run **dtr** on a remote node, you must run it on a default nonprivileged account (guest account), because you cannot specify access-control information with this qualifier.

/print or **/noprint** (default)	tells **dts** whether to print (log) test results.
/statistics or **/nostatistics** (default)	tells **dts** whether to print statistics on data and interrupt tests.
/display or **/nodisplay** (default)	tells **dts** whether to print the data and interrupt messages transmitted to **dtr**.
/speed=*number*	specifies the test line speed in bits per second (default=0); **dts** uses this data for reporting statistics.
test-specific-qualifiers	specifies any number of test-specific qualifiers, as defined in the following sections. Test-specific qualifiers apply to the current test only.

The command syntax described in this section uses the following conventions:

- All test names and qualifiers can be abbreviated to the first three or more unique characters.
- The default values for a qualifier remain in effect until a different value is specified. The specified value then becomes the new default for all following tests until that value is changed.

NOTE

Be sure to review the graphic conventions described in the Preface.

5.5.1.1 Connect Test Syntax

Use the following format to perform a **connect** test:

connect[*qualifiers*][*test-specific-qualifiers*]

where *test-specific-qualifiers* can be any of the following:

/type=*subtest*	specifies the type of test, where *subtest* can be:	
	accept	connect accept test (default)
	reject	connect reject test
/return=*type* or **noreturn**	specifies the type of data returned by **dtr**, where *type* can be:	
	standard	standard user data
	received	received user data
	noreturn	causes no optional user data to be returned

EXAMPLE:

This command invokes a **connect accept** test (by default) with remote node MONTRL.

```
dts> connect/nodename=montrl/return=received RET
```

The **dtr** program returns received user data as part of the test.

5.5.1.2 Disconnect Test Syntax

Use the following format to perform a **disconnect** test:

disconnect[*qualifiers*] [*test-specific-qualifiers*]

where *test-specific-qualifiers* can be any of the following:

type=subtest	specifies the type of test, where *subtest* can be:
	synchronous synchronous disconnect test
	abort disconnect abort test (default)
/**return**=*type* /**noreturn**	specifies the type of data returned by **dtr**, where *type* can be:
	standard standard user data
	received received user data

The /**noreturn** qualifier causes no optional user data to be returned.

EXAMPLE:

This command invokes a **synchronous disconnect** test with remote node PARIS.

```
dts> disconnect/nodename=paris/type=synchronous RET
```

The **dtr** program will not return any optional user data.

5.5.1.3 Data Test Syntax

Use this format to perform a **data** test:

data[*qualifiers*] [*test-specific-qualifiers*]

where *test-specific-qualifiers* can be any of the following:

/**type**=*subtest*	specifies the type of test, where *subtest* can be:
	sink sink test (default)
	sequence sequence test
	pattern pattern test
	echo echo test
/**size**=*number*	specifies data message length in bytes, where *number* is a value from 1 to 2,048 (default=128). NOTE: The minimum value for *number* on a sequence test is 4 and on a pattern test is 5.
/test-duration	specifies duration of the test in one of the following formats:
	seconds=*number* range: 1 to 60
	minutes=*number* range: 1 to 60
	hours=*number* range: 1 to 24
	The default is **seconds**=15.
/**flow**=*type* or /**noflow** (default)	specifies type of flow control if any where *type* can be:
	segment segment flow control
	message message flow control (default, if /**flow** is specified)
	If **dtr** is running on DECnet–ULTRIX software, it must use the system default.
/**rqueueD**=*number*	specifies number of pending receives for **dtr** to maintain, where *number* is a value from 1 to 16 (default = 1). If the remote system is a DECnet–ULTRIX node, this parameter is ignored.

/nak=*number* or **/noack**
: specifies the number of segments between negative acknowledgments (NAKs). If the remote system is a DECnet–ULTRIX node, this parameter is ignored.

/back=*number* or */noback*
: specifies the number of segments before back pressuring. If the remote system is a DECnet–ULTRIX node, this parameter is ignored.

EXAMPLE:

This command invokes the **data** test with the sink subtest (by default). The **dts** program sends messages to **dtr** on node JONES (by default from a previous command). The message size is 512 bytes, and the duration of the test is 30 seconds.

```
dts> data/size=512/seconds=30 RET
 DTS --I-- Test started at 11:23:30
 DTS --I-- Test finished at 11:24:00

Test parameters:
     Target node          "jones"
     Test duration (sec)  30
     Message size (bytes) 512)

Summary statistics:
     Total messages SENT  48
     Total bytes SENT     24576
     Messages per second  1.60
     Bytes per second     819.20
     Line throughput (baud)  6553
```

5.5.1.4 Interrupt Test Syntax

Use this format to perform an **interrupt** test:

interrupt[*qualifiers*] [*test-specific-qualifiers*]

where *test-specific-qualifiers* can be any of the following:

/type=*subtest*
: specifies the type of test, where subtest can be:

  sink sink test (default)

  sequence sequence test

  pattern pattern test

  echo echo test

/size=*number*
: specifies data message length in bytes, where *number* is a value from 1 to 16 (default = 16). NOTE: The minimum value for *number* on a sequence test is 4 and on a pattern test is 5.

test-duration
: specifies duration of the test in one of the following formats:

  seconds=*number* range: 1 to 60

  minutes=*number* range: 1 to 60

  hours=*number* range: 1 to 24

  The default is **seconds**=15.

/rqueue=*number*
: specifies number of pending receives for **dtr** to maintain, where *number* is a value from 1 to 16 (default=1). If the remote system is DECnet–ULTRIX, this parameter is ignored.

EXAMPLE:

This command invokes the **interrupt** test with the pattern subtest. The **dts** program sends interrupt messages to **dtr** on node DALLAS, where test information is to be printed. The default is used for message size, and the duration of the test is 30 seconds.

```
dts>interrupt/nodename=dallas/print/type=pat/seconds=30 RET

 DTS --I-- Test started at 17:44:10
 DTS --I-- Test finished at 17:44:40

Test parameters:

Target node                "dallas"
Test duration (sec)        30
Message size (bytes)       16

Summary statistics:
Total messages SENT        2734
Total bytes SENT           43744
Messages per second        91.1
Bytes per second           1458
Line throughput (baud)     11665
```

Chapter 6

DECnet–ULTRIX Network Maintenance Commands

This chapter describes two DECnet–ULTRIX maintenance commands. The format for this information corresponds to that in the ULTRIX reference pages. See the *ULTRIX Reference* manuals for more information on format.

The name of each command being described appears in a running head, followed by the appropriate section number and suffix in parentheses. For example, **dts(8dn)** appears on the reference pages that describe **dts**. The number **8** indicates that the section describes maintenance commands. The **dn** suffix indicates that the commands are used in the DECnet domain.

The command descriptions use the graphic conventions described in the Preface.

Table 6-1 summarizes the functions of the DECnet–ULTRIX maintenance commands.

Table 6–1: DECnet–ULTRIX Maintenance Commands

Command	Function
dts	Evokes the DECnet Test Sender.
ncp	Runs the Network Control Program.

The following command descriptions also appear on-line in the **dts(8dn)** and **ncp(8dn)** manual pages.

dts (8dn)

NAME

dts — evoke the DECnet test sender

SYNTAX

dts [**test**[*/qualifiers*][*/test-specific-qualifiers*]]

where

test is the name of the test you want to run:

connect	specifies a connect test. These tests verify that the **dtr** node can process connection requests and return **accept** and **reject** messages with or without optional data.
data	specifies a data test. These tests include sink, sequence, pattern, and echo tests.
disconnect	specifies a disconnection test. These tests verify that the node can send out disconnection messages with or without optional data.
interrupt	specifies an interrupt test. These tests include sink, sequence, pattern, and echo tests for interrupt data.

qualifiers are the valid qualifiers for this command. Once specified, these qualifiers remain in effect for all applicable tests until you either change them or exit from **dts**. Precede each qualifier with a slash (/). You can specify any number of these qualifiers:

nodename=*node-id*	specifies the name or address of the DECnet node on which you want **dtr** to run. The default is the local node. If you run **dtr** on a remote node, you must run it on a default nonprivileged account (guest account) because you cannot specify access-control information with this qualifier.
print or **noprint** (default)	tells **dts** whether or not to print test results.
statistics or **nostatistics** (default)	tells **dts** whether or not to print statistics on data and interrupt tests.
display or **nodisplay** (default)	tells **dts** whether or not to print the data and interrupt message transmitted to **dtr**.
speed=*number*	specifies the test line speed in bits per second (default = 0). The **dts** program uses this data for reporting statistics.

test-specific-qualifiers

are the valid qualifiers for the test you specify. Test-specific qualifiers apply only to the current test.

DESCRIPTION

The **dts** utility is the DECnet–ULTRIX transmitter test program that runs four tests: connect, data, disconnect, and interrupt.

Specify the test you want in one of two ways:

% dts *test[/qualifiers][/test-specific-qualifiers]

or

% dts
dts> test[/qualifiers][/test-specific-qualifiers]

The **dts** program initiates each test by issuing a connect request to the **dtr** program. Parameter information associated with each type of test requested is passed by **dts** to **dtr** during a connection.

EXAMPLE

The following command invokes a data test with remote node OHLONE. Data messages of 512 bytes are transmitted to the **dts** program on the remote node, which then echos them back. The test runs for 10 seconds. Test parameters and summary statistics are displayed after the test completes.

```
% dts data/nodename=ohlone/type=echo/size=512/seconds=10 RET
```

ncp (8dn)

NAME

ncp — run the Network Control Program

SYNTAX

ncp [*command*] [*component*] [*parameter list*]

where

command specifies the command you want to execute:

help [*command...*]	displays a short description of the command specified in the argument list. If no arguments are given, it displays a list of the recognized commands.
show	displays information about the volatile database.
list	displays information about the permanent database.
set	creates or modifies parameters in the volatile database. Also specifies a new node as the executor.
define	creates or modifies parameters in the permanent database.
clear	removes parameters from the volatile database.
purge	removes parameters from the permanent database.
zero	resets the DECnet counters, including line and circuit counters.
tell	sends **ncp** commands to a remote node for execution.
loop	tests either a node in the network or a circuit on the executor node.

For a complete list of **ncp** commands and their use in a DECnet–ULTRIX environment, see the *DECnet–ULTRIX NCP Command Reference* manual.

DESCRIPTION

The DECnet–ULTRIX Network Control Program (**ncp**) is a network management utility. You can use **ncp** commands to configure, control, monitor, and test DECnet nodes.

To execute **ncp**, use one of the following formats:

% **ncp** *command*

or

% **ncp**
ncp> *command*

RESTRICTION

You must have superuser privileges to use an **ncp** command that modifies a database. However, any user can use the **ncp help**, **show**, and **list** commands to display information from the permanent and volatile databases.

EXAMPLE

The following example starts the DECnet–ULTRIX software running on your local system:

```
% ncp set executor state on RET
```

The following example sends the **show executor characteristics** command to the remote DECrouter 200 node named ROUTER, where the command is then executed. The information about ROUTER is displayed at the local node.

```
% ncp tell router show executor characteristics RET
```

Index

A

C

D

E

F

Q

R

S

T

U

V

W

HOW TO ORDER ADDITIONAL DOCUMENTATION

DIRECT TELEPHONE ORDERS

In Continental USA
call 800–DIGITAL

In Canada
call 800–267–6215

In New Hampshire
Alaska or Hawaii
call 603–884–6660

In Puerto Rico
call 809–754–7575

ELECTRONIC ORDERS (U.S. ONLY)

Dial 800–DEC–DEMO with any VT100 or VT200
compatible terminal and a 1200 baud modem.
If you need assistance, call 1–800–DIGITAL.

DIRECT MAIL ORDERS (U.S. and Puerto Rico*)

DIGITAL EQUIPMENT CORPORATION
P.O. Box CS2008
Nashua, New Hampshire 03061

DIRECT MAIL ORDERS (Canada)

DIGITAL EQUIPMENT OF CANADA LTD.
940 Belfast Road
Ottawa, Ontario, Canada K1G 4C2
Attn: A&SG Business Manager

INTERNATIONAL

DIGITAL
EQUIPMENT CORPORATION
A&SG Business Manager
c/o Digital's local subsidiary
or approved distributor

Internal orders should be placed through the Software Distribution Center (SDC),
Digital Equipment Corporation, Westminster, Massachusetts 01473

*Any prepaid order from Puerto Rico must be placed
with the Local Digital Subsidiary:
809–754–7575

DECnet–ULTRIX
Network Management
AA–EE38D–TE

READER'S COMMENTS

What do you think of this manual? Your comments and suggestions will help us to improve the quality and usefulness of our publications.

Please rate this manual:

	Poor				Excellent
Accuracy	1	2	3	4	5
Readability	1	2	3	4	5
Examples	1	2	3	4	5
Organization	1	2	3	4	5
Completeness	1	2	3	4	5

Did you find errors in this manual? If so, please specify the error(s) and page number(s).

_____

_____

_____

_____

General comments:

_____

_____

_____

_____

Suggestions for improvement:

_____

_____

_____

_____

Name _____ Date _____

Title _____ Department _____

Company _____ Street _____

City _____ State/Country _____ Zip _____

DO NOT CUT – FOLD HERE AND TAPE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY LABEL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

digital™

Networks and Communications Publications
550 King Street
Littleton, MA 01460–1289

DO NOT CUT – FOLD HERE

CUT ON THIS LINE